B: Amendments to The Claims:

Amend the claims to read as follows:

Claim 1. (Currently Amended) A multiprocessor computer system, comprising:

a plurality of processing nodes and a plurality of dynamic cache coherency regions using caches associated with said processing nodes, and having a supervisor software and

cache controller logic in said processing nodes controlling software-initiated movement of software processes between said plurality of cache coherency regions to delay any need for purging by changing coherency mode bits without immediately requiring a selective purging of cache contents in one or more of said processing nodes that are part of a chache coherency region being changed and when moving a software process between two distinct sets of processing nodes a cache line cannot be marked as shared in two separate coherency regions, and when said supervisor software program is moving a coherency region from one distinct set of old processing nodes to another distinct set of new processing nodes it is ~~effectively~~ leaving behind old cache entries for the coherency region on ~~the~~ old nodes and ensures that these old cache entries will be seen by incoming storage requests originating from ~~the~~ new processing nodes and that cache entries for ~~the same~~ main storage addresses ~~will not be established~~ in the new processing nodes will not be established the old entries are invalidated.

Claim 2. (Currently Amended) The multiprocessor computer system according to claim 1, including supervisor software which enables said cache controller logic in a processing node associated with a processor making a storage request to be sure upon an incoming storage request of a processor for a storage address that no copy of the requested storage address exists outside that processor’s current coherency region, as specified by the setting of mode bits for a current coherency region mode, whenever a cache entry for a requested storage address is found to exist on any cache in any of said plurality of said processing nodes ~~that contains the processor that initiated said incoming storage request~~ outside that processor’s current coherency region.

Claim 3. (Currently Amended) The multiprocessor computer system according to claim 2, wherein said supervisor software creates a unique Coherency Region ID for each process in said multiprocessor computer system that has its own coherency region.

Claim 4. (Currently Amended) The multiprocessor computer system according to claim ~~2~~ 3, wherein supervisor software creates a table in said multiprocessor computer system for each processing node in the system which has an entry for every Coherency Region ID of a process that is currently allowed to be dispatched on ~~said~~ a processing node on which a process is running.

Claim 5. (Currently Amended) The multiprocessor computer system according to claim 2, wherein said supervisor software creates a unique Coherency Region ID for each process of a multiprocessor system that has its own coherency region and one or more coherency mode bits for each processor node in the multiprocessor computer system, and said coherency mode bits and coherency region ID associated with a processor are sent together with each storage transaction that is initiated by that processor when the transaction is transmitted for communication to another processor of said multiprocessor computer system.

Claim 6. (Currently Amended) The multiprocessor computer system according to claim 2, wherein said supervisor software creates a unique Coherency Region ID for each process of a multiprocessor system that has its own coherency region and one or more coherency mode bits for each processor node in the multiprocessor system to identify a coherency region to a node controller for a processing node by said Unique Coherency Region ID.

Claim 7. (Currently Amended) The multiprocessor computer system according to claim 2, wherein said supervisor software creates a unique Coherency Region ID for each process of a multiprocessor system that has its own coherency region and one or more coherency mode bits for each processor node in the multiprocessor computer system, and wherein said mode bits associated with each transaction are used to determine which caches must participate in any storage transactions that they receive from any of the processors of said multiprocessor computer system.

Claim 8. (Currently Amended) The multiprocessor computer system according to claim 2, wherein said supervisor software creates a unique Coherency Region ID for each process of a multiprocessor system that has its own coherency region and one or more coherency mode bits for each processor node in the multiprocessor computer system and enables multiple cache coherency regions to operate without the use of cache purges during some operations which move software processes between coherency regions.

Claim 9. (Currently Amended) The multiprocessor computer system according to claim 8, wherein said supervisor software moves a software process out of one first coherency region that is no longer going to be used by said software process and into another coherency region that has been created to cover the same address space as the one first coherency region but which will include a new set of processing nodes.

Claim 10. (Original) The multiprocessor computer system according to claim 2, wherein said supervisor software creates a unique Coherency Region ID for each process that has its own coherency region and moves a software process from one coherency region encompassing one set of processing nodes to another coherency region encompassing another set of processing nodes without requiring cache purges of caches in any of the processing nodes.

Claim 11. (Currently Amended) The multiprocessor computer system according to claim 10, wherein ~~if~~ moving a software process to said another coherency region which contains fewer hardware processing nodes than the original coherency region reduces the effective ~~, the~~ size of the coherency region for said processing nodes~~ has been effectively been reduced~~.

Claim 12. (Currently Amended) The multiprocessor computer system according to claim 1, wherein said multiprocessor computer system having a plurality of said processing nodes uses ~~a table of~~ active coherency region information, which active coherency region information is associated with each processing node to determine when to alter the processing nodes' cache state transitions.

Claim 13. (Currently Amended) The multiprocessor computer system according to claim 12, wherein a supervisor software initializes ~~said tables~~ a table having active coherency region information associated with each processing node and an entry in said table is made for each coherency region that the supervisor software intends to use on ~~that~~ an associated processing node .

Claim 14. (Original) The multiprocessor computer system according to claim 1, wherein a supervisor software assigns a unique coherency region ID for each coherency region which the supervisor can associate with all software processes that access storage addresses that are encompassed by the coherency region.

Claim 15. (Original) The multiprocessor computer system according to claim 1, wherein processing nodes are able to identify incoming storage requests which target lines that are no longer part of the address space of any software process that is currently enabled by the supervisor software to be dispatched on the node.

Claim 16. (Original) The multiprocessor computer system according to claim 1, wherein processing nodes are able to identify incoming storage requests which target lines that are no longer part of the address space of any software process that is currently enabled by the supervisor software to be dispatched on the node to thereby identify cache lines that are no longer actively used by any software processes on that processing node and to change the cache entries for that processing node to invalid in response to a storage request from outside the coherency region.

Claim 17. (Original) The multiprocessor computer system according to claim 1, wherein processing nodes are able to identify incoming storage requests which target lines that are no longer part of the address space of any software process that is currently enabled by the supervisor software to be dispatched on the node and allows all of the caches in said multiprocessor computer system to continue processing coherency transactions while the coherency boundaries for a software process are effectively changed.

Claim 18. (Original) The multiprocessor computer system according to claim 1, wherein processing nodes are able to identify incoming storage requests which target lines that are no longer part of the address space of any software process that is currently enabled by the supervisor software to be dispatched on the node such that cache lines belonging to a software process that is no longer actively being dispatched on a given processing node are identified and invalidated thereby enabling their reuse.

Claim 19. (Original) The multiprocessor computer system according to claim 1, wherein a supervisor software uses processor state information to determine which caches in the multiprocessor computer system are required to examine a coherency transaction produced by a single originating processor’s incoming storage request.

Claim 20. (Previously Amended) The multiprocessor computer system according to claim 19, wherein a processing node of the multiprocessor computer system has dynamic coherency boundaries such that the multiprocessor computer system uses only a subset of the total processors in a system for a single workload at any specific point in time and can optimize the cache coherency as the supervisor software expands and contracts the number of processors which are being used to run any single workload.

Claim 21. (Original) The multiprocessor computer system according to claim 20, wherein multiple instances of processing nodes can be connected with a second level controller to create a large multiprocessor system.

Claim 22. (Currently Amended) The multiprocessor computer system according to claim 21, wherein said supervisor software creates a unique Coherency Region ID for each process of a multiprocessor system that has its own coherency region and one or more coherency mode bits for each processor node in the multiprocessor computer system and enables multiple cache coherency regions to operate without the use of cache purges during some operations which move software processes between coherency regions and a node controller uses said mode bits to determine which processors must receive any given transaction that is received by the node controller.

Claim 23. (Original) The multiprocessor computer system according to claim 22, wherein a second level controller uses the mode bits to determine which processing nodes must receive any given transaction that is received by the second level controller.

Claim 24. (Currently Amended) The multiprocessor computer system according to claim 21, wherein said ~~supervision~~ supervisor software uses logical partitions which are mapped to allowable physical processors and a distinct cache coherency region can be defined for each partition using a hypervisor.

Claim 25. (Currently Amended) The multiprocessor computer system according to claim ~~2~~ 22, wherein said coherency region ID is used to perform the function of a cache coherency mode and a node controller determines which physical processing nodes are associated with specific coherency region Ids.

Claim 26. (Original) The multiprocessor computer system according to claim 3, wherein any incoming storage request which misses all of the caches in an originator’s coherency region is then sent on to all processing nodes in the entire system, regardless of the setting of said mode bits.

Claim 27. (Currently Amended) The multiprocessor computer system according to claim 3, wherein any incoming storage ~~request which~~ requests which hit in an originator’s coherency region but which do not have a correct cache state do not need to be sent outside the coherency region.

Claim 28. (Previously Amended) A method for use in a multiprocessor computer system, comprising the steps of:

moving software processes between a plurality of cache coherency regions for caches associated with a plurality of processing nodes of said multiprocessor computer system without requiring a selective purging of cache contents in one or more of said processing nodes, after supervisor software creates a unique Coherency Region ID for each process that has its own coherency region, and

said supervisor software creates a table ~~for each processing node~~ in the multiprocessor computer system for each processing node which table has an entry for every Coherency Region ID that is currently allowed to be dispatched on said processing node.

Claim 29. (Currently Amended) The method for the multiprocessor computer system according to claim 28, wherein ~~said~~ a unique Coherency Region ID for each process and coherency mode bits ~~and coherency region ID~~ associated with a processor node initiating a storage request are sent together with each storage transaction that is initiated by that processor node initiating a storage request with a requested storage address when the transaction is transmitted for communication to another processor of said multiprocessor computer system.

Claim 30. (Currently Amended) The method for the multiprocessor computer system according to claim 28, including a step of enabling with supervisor software the multiprocessor computer system's cache controller logic in a processing node to respond to current coherency region mode bit settings ~~be sure that~~, upon receipt at said processing node ~~that~~ of an incoming storage request for a storage address, when no copy of the requested storage address exists outside ~~said~~ a processing node's current coherency region, as specified by a current coherency region mode bit settings, whenever a cache entry for a requested storage address is found to exist on any cache in any of said multiprocessor computer system's processing nodes that contains a processor that initiated said incoming storage request.

Claim ~~27~~ 31. (RENUMBERED AND CANCELED)